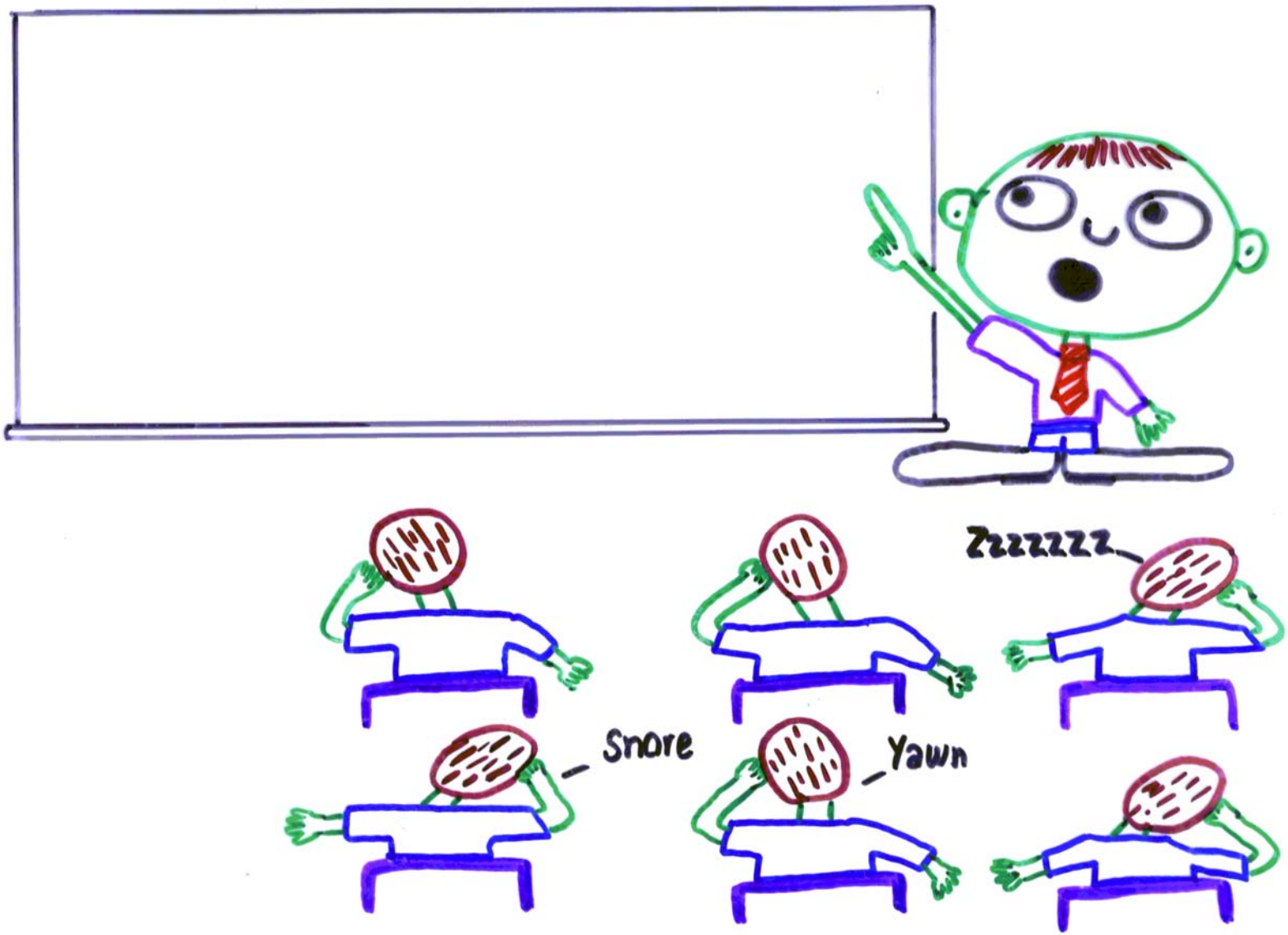
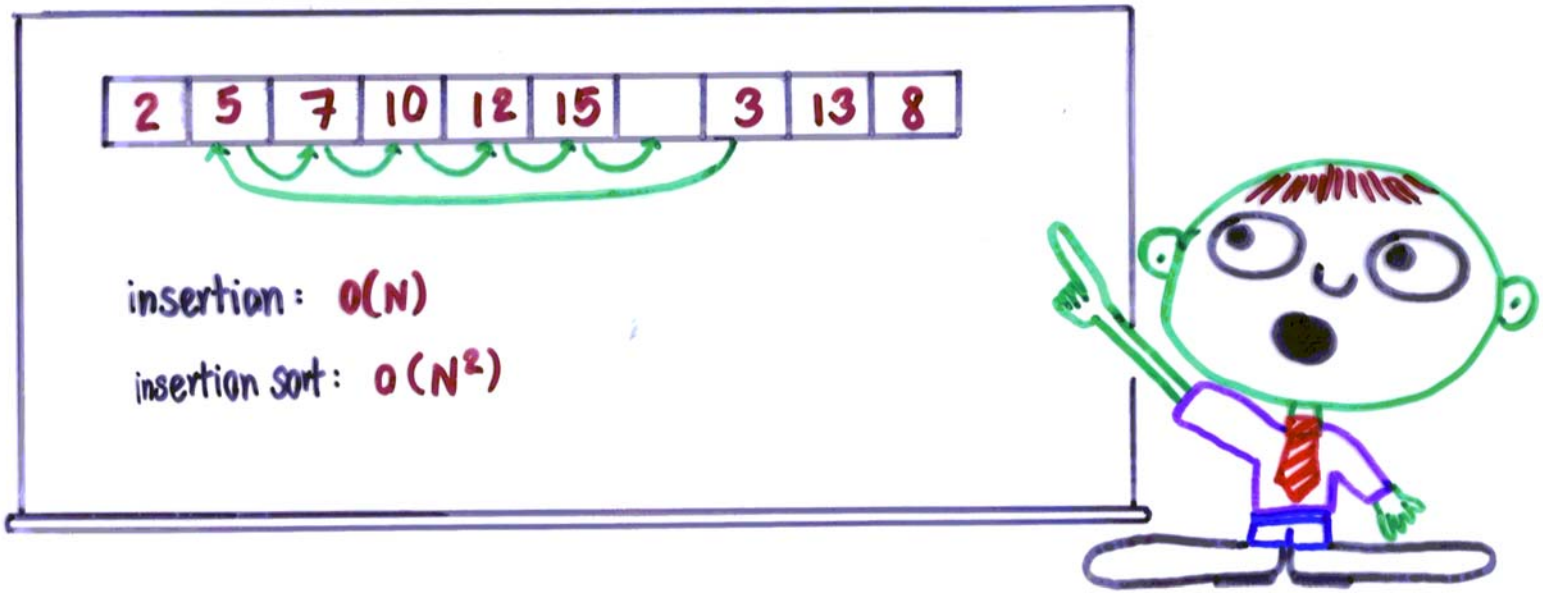
My Relationship with Insertion Sort

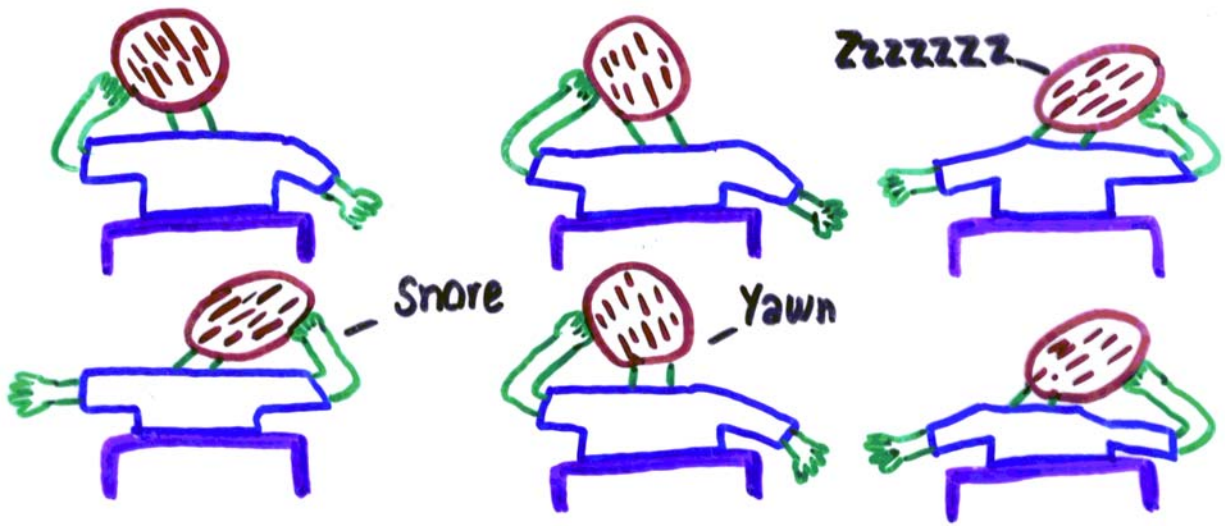
Fall 1990. I'm an undergraduate.



My Relationship with Insertion Sort

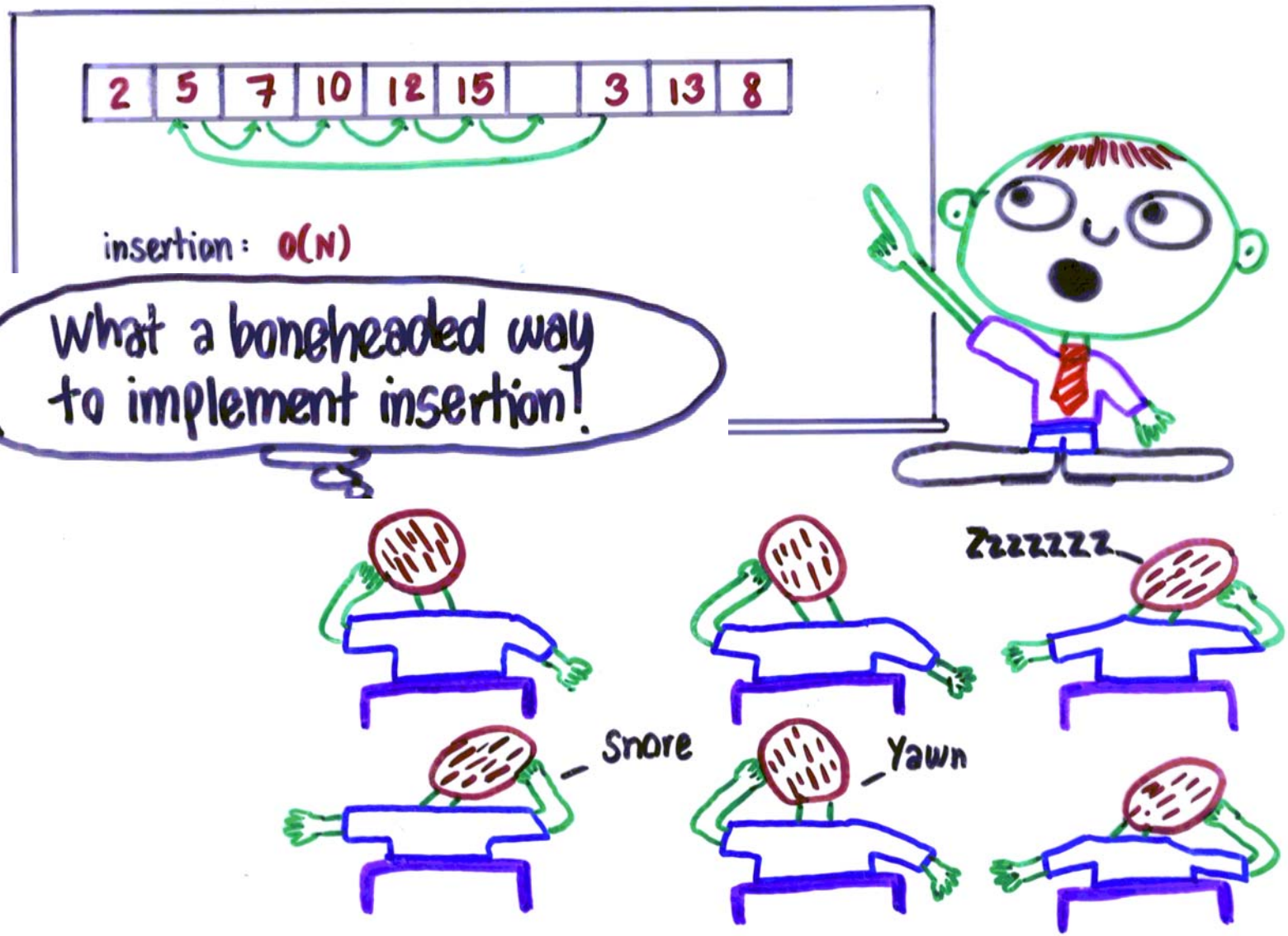
Fall 1990. I'm an undergraduate.





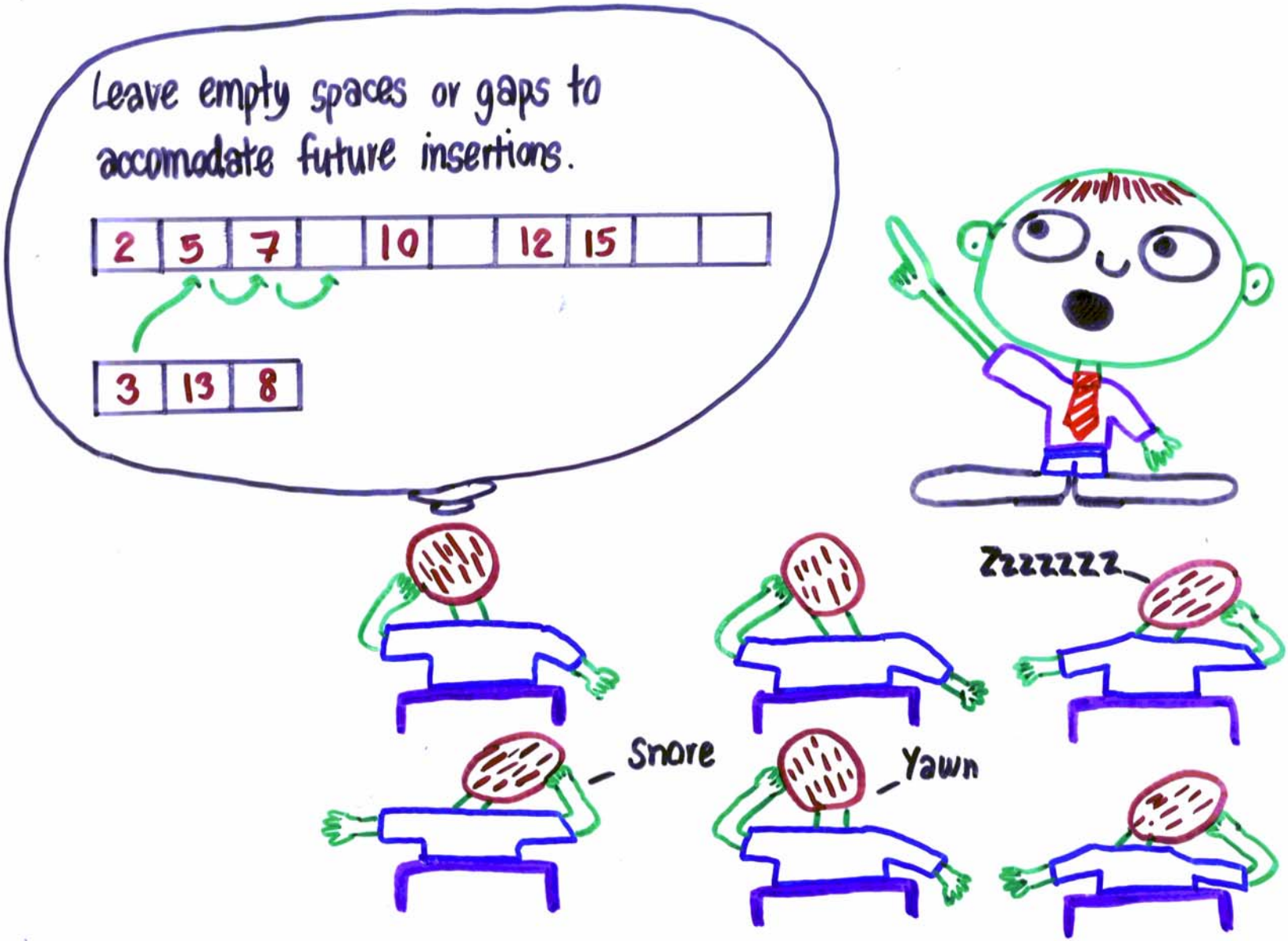
My Relationship with Insertion Sort

Fall 1990. I'm an undergraduate.



My Relationship with Insertion Sort

Fall 1990. I'm an undergraduate.



My Relationship with Insertion Sort

Anybody who has spent time in a library knows that insertions are cheaper than linear time.

Everybody (except computer scientists) know that gaps make inserts cheaper than $O(n)$ ("folk computer science").

Question: how much.

Small library

big library

14 years after that lecture on insertion sort

(1 BA, 1 DEA, 1 Magistère, 1 Ph.D., tenure-almost)...

Insertion Sort is $O(N \log N)$

Michael A. Bender

Martin Farach-Colton

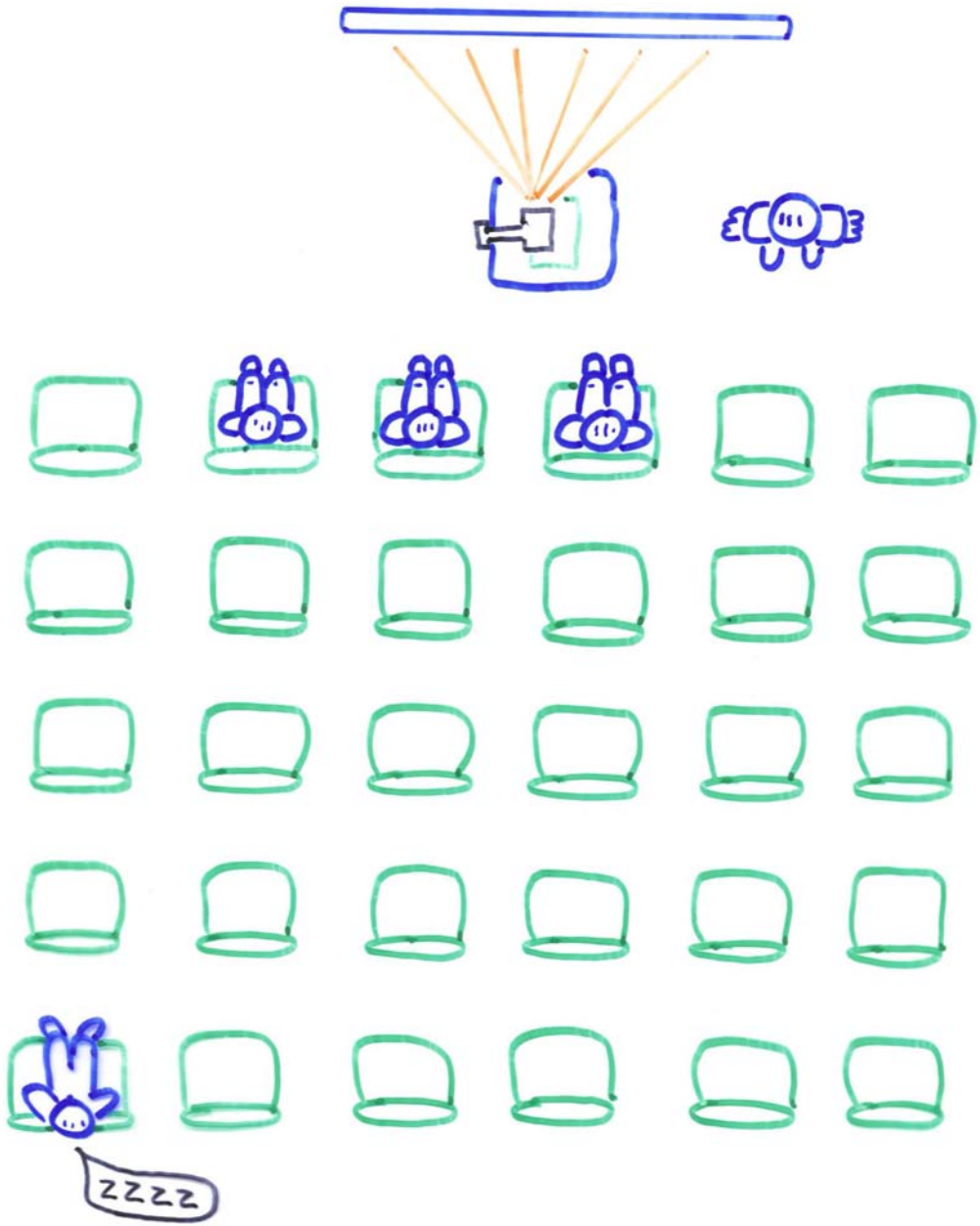
Miguel Mosteiro

Results

- **LibrarySort**, a natural implementation of **InsertionSort** with gaps.

- **Theorem: LibrarySort** runs in $O(N \lg N)$ time w.h.p. and uses linear space. Each insertion is exp $O(1)$ and $O(\lg N)$ w.h.p..

Overview of Talk



Library Sort is $O(N \log N)$

for $k = 1$ to N do

find location to insert x_k (binary search)

insert x_k

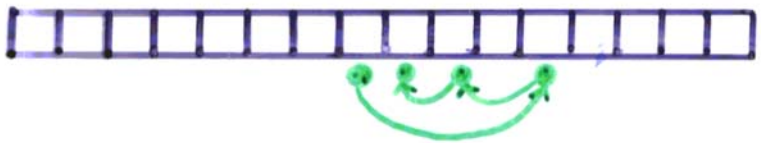
if k is power of 2 then

rearrange elements evenly in $(2+\epsilon)k$ - sized region.

Library Sort is $O(N \log N)$

for $k = 1$ to N do

find location to insert x_k (binary search)



$O(\log N)$

insert x_k

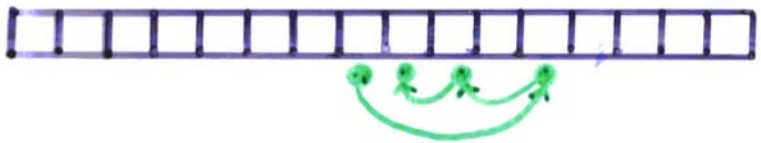
if k is power of 2 then

rearrange elements evenly in $(2+\epsilon)k$ - sized region.

Library Sort is $O(N \log N)$

for $k = 1$ to N do

find location to insert x_k (binary search)



$O(\log N)$

insert x_k

$O(\log N)$ w.h.p.

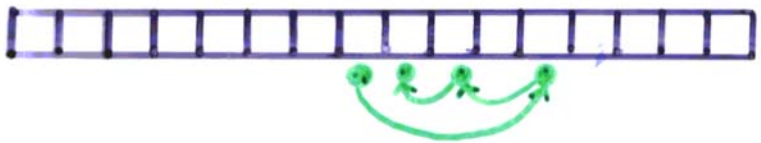
if k is power of 2 then

rearrange elements evenly in $(2+\epsilon)k$ - sized region.

Library Sort is $O(N \log N)$

for $k = 1$ to N do

find location to insert x_k (binary search)



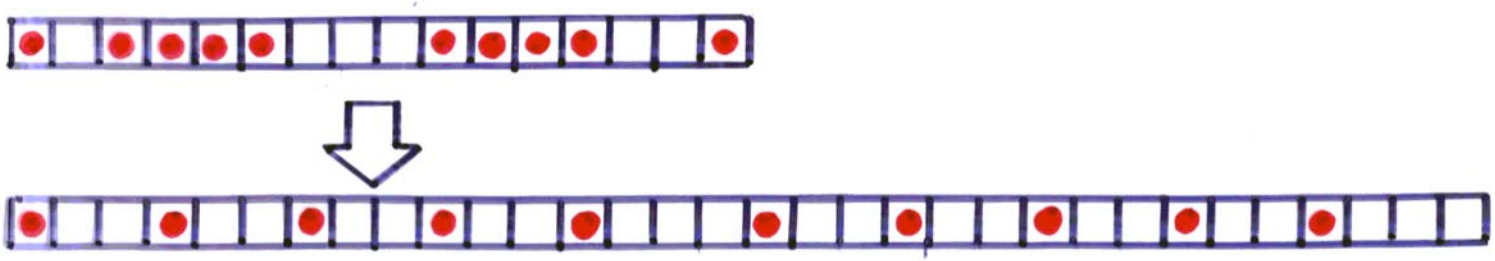
$O(\log N)$

insert x_k

$O(\log N)$ w.h.p.

if k is power of 2 then

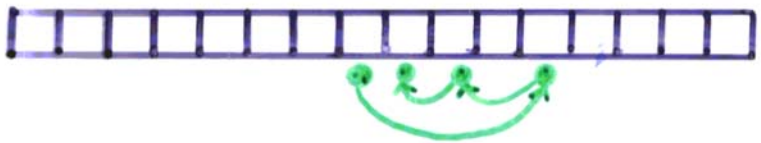
rearrange elements evenly in $(2+\epsilon)k$-sized region.



Library Sort is $O(N \log N)$

for $k = 1$ to N do

find location to insert x_k (binary search)



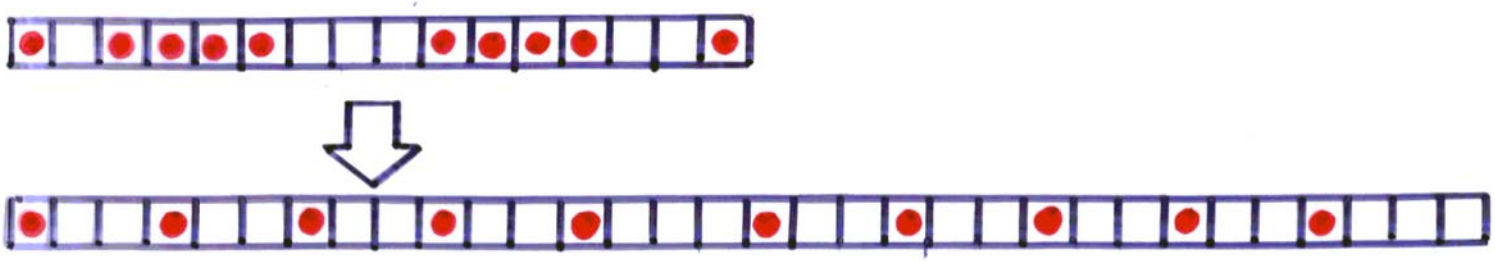
$O(\log N)$

insert x_k

$O(\log N)$ w.h.p.

if k is power of 2 then

rearrange elements evenly in $(2+\epsilon)k$-sized region.



Thm: each insertion has cost $O(\lg N)$ w.h.p.

Thm: each insertion has cost $O(\lg N)$ w.h.p.

Pf idea: Phase ℓ: elements $2^\ell \longrightarrow 2^{\ell+1}$ inserted.

beginning of phase: elements are rebalanced (evenly spread in array).

2^ℓ support elements

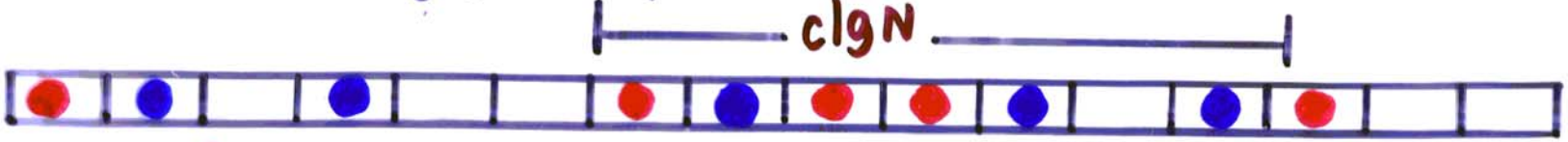
Thm: each insertion has cost $O(\lg N)$ w.h.p.

Pf idea: Phase ℓ: elements $2^\ell \longrightarrow 2^{\ell+1}$ inserted.

beginning of phase: elements are rebalanced (evenly spread in array).

2^ℓ support elements

end of phase: for sufficiently large constant c, any region of size $c \lg N$ has gaps w.h.p.

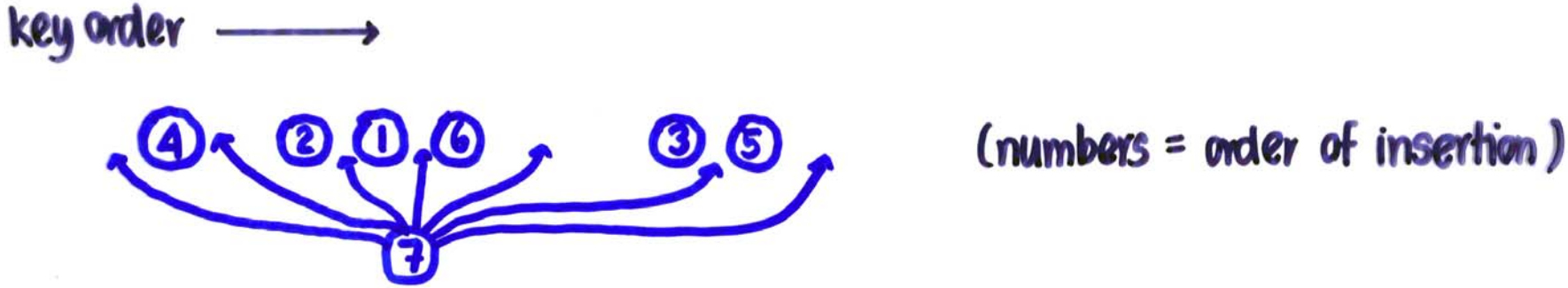


2^ℓ support and 2^ℓ intercalated elements

Invariant

The (k+1)st element is equally likely to be inserted between any two of the k elements already in the array.

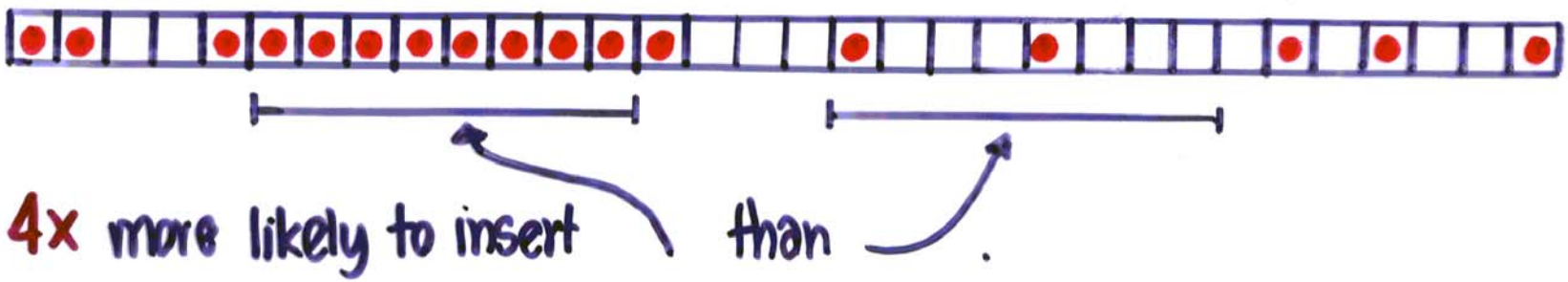
Follows because elements are inserted in random order.



(numbers = order of insertion)

Difficulty

Dense regions of array act as attractors.

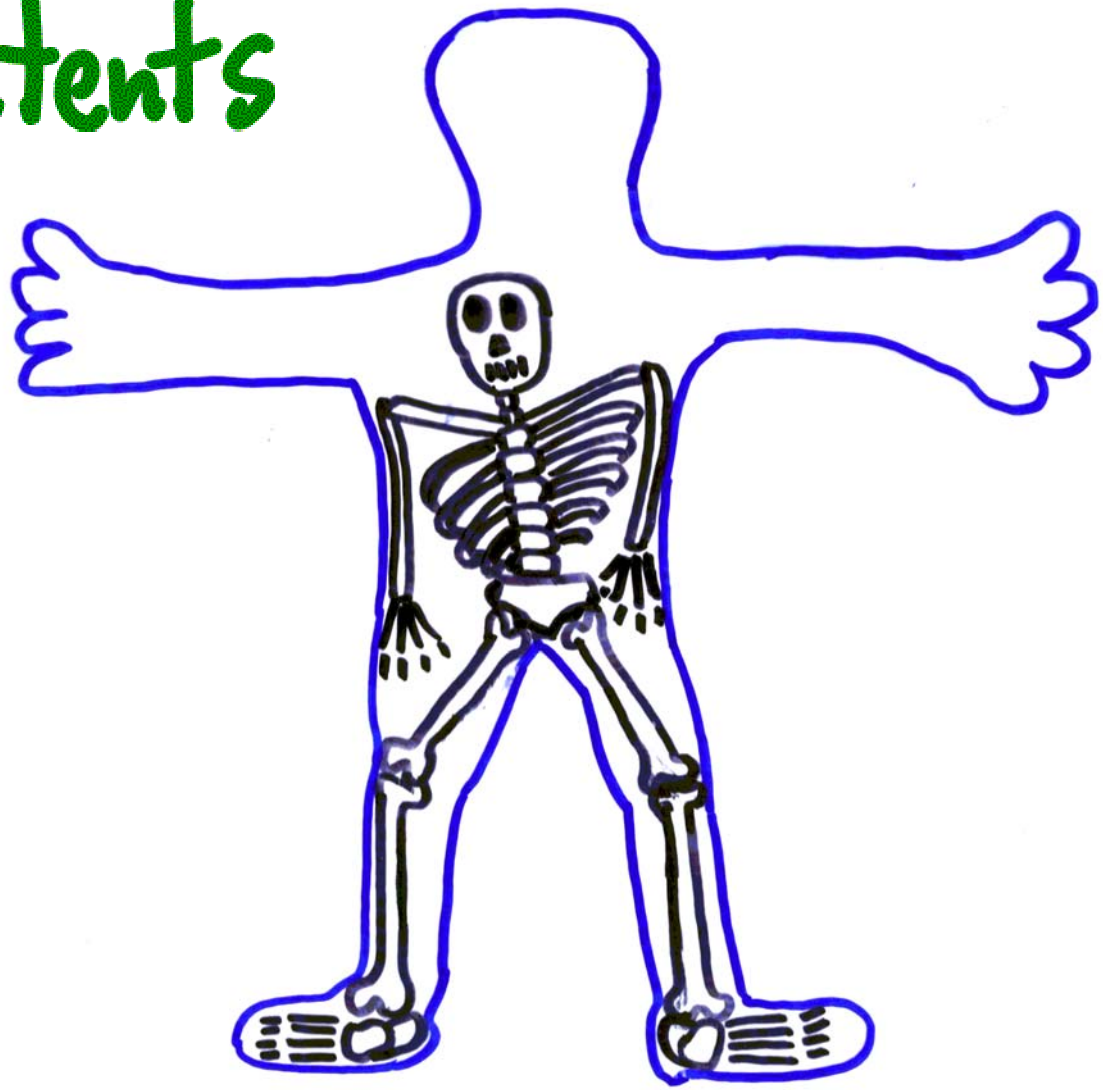


4x more likely to insert than .

Need to show that despite attraction dense regions do not get too big.

Contents

Contents



Contents may have settled during shipping.

Balls and Bins Game

Idea: model attracting regions when M elmts in array.

M additional balls thrown in bins.

k^{th} ball thrown into Bin A or B with probability proportional to # balls in bins

$$X_k = \begin{cases} 1 & \text{if ball } k \text{ thrown into Bin A} \\ 0 & \text{otherwise.} \end{cases}$$

Balls and Bins Game

Idea: model attracting regions when M elmts in array.

M additional balls thrown in bins.

k^{th} ball thrown into Bin A or B with probability proportional to # balls in bins

$$X_k = \begin{cases} 1 & \text{if ball } k \text{ thrown into Bin A} \\ 0 & \text{otherwise.} \end{cases}$$

Thm: Number of balls thrown in Bin A is

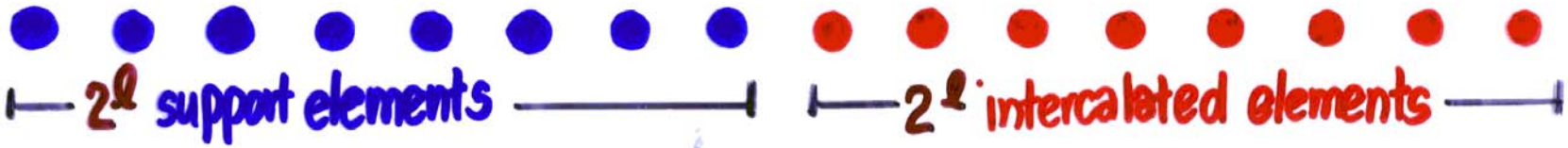
$$X = X_{M+1} + X_{M+2} + \dots + X_{2M} = O(\lg N).$$

Issue: Random variables $X_{M+1} \dots X_{2M}$ are positively correlated.

Need an alternative approach

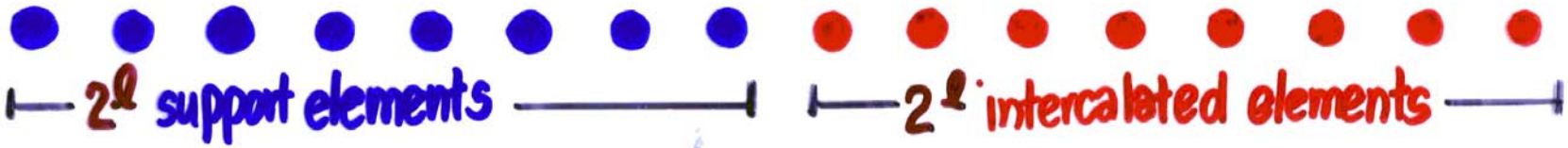
Alternative Analysis

Elements ordered by insertion order : random permutation on keys



Alternative Analysis

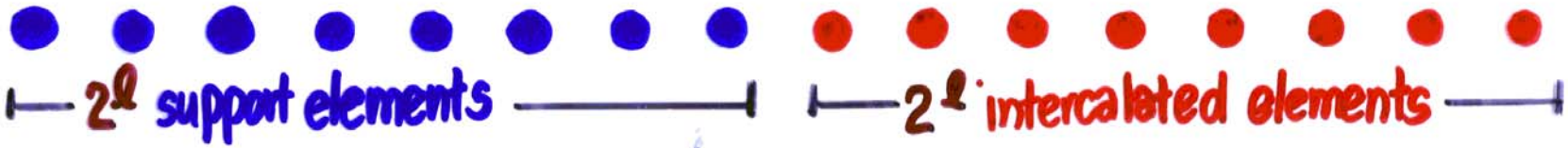
Elements ordered by insertion order : random permutation on keys



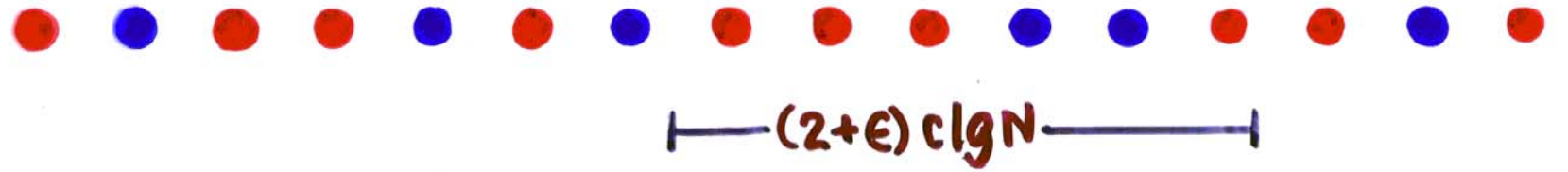
Elements ordered by keys : random permutation on insert order

Alternative Analysis

Elements ordered by insertion order : random permutation on keys



Elements ordered by keys : random permutation on insert order

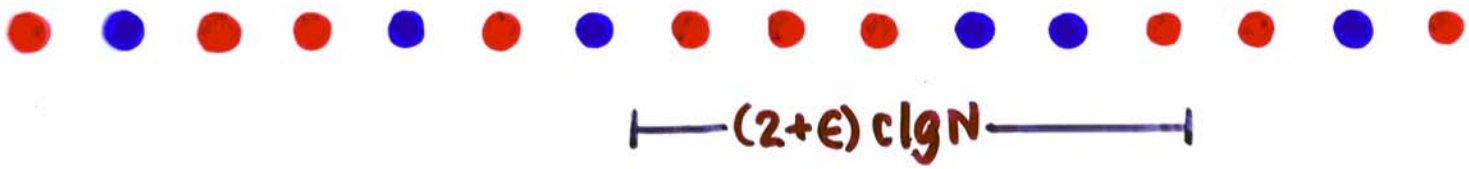


Claim: In any window of size $\Theta(\lg N)$ there are $\Theta(\lg N)$ support elements and $\Theta(\lg N)$ intercalated elements w.h.p.

$\Rightarrow$ evenly distributed.

Alternative Analysis

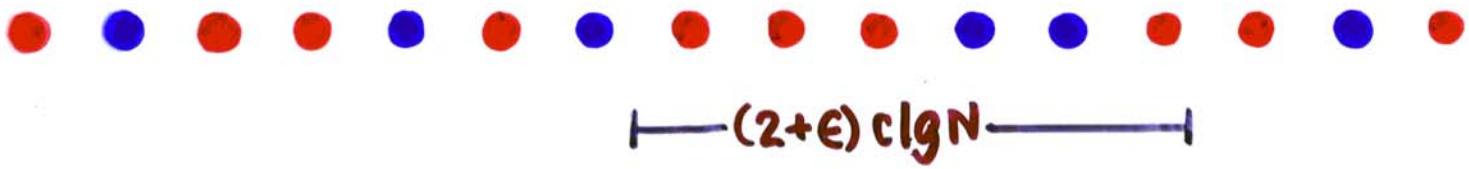
Elements ordered by keys: random permutation on insert order



Claim: For sufficiently large c, in any window of size $(2+\epsilon)c \lg N$, there are $> c \lg N$ support elements and $< (1+\epsilon) c \lg N$ intercalated elements.

Alternative Analysis

Elements ordered by keys: random permutation on insert order

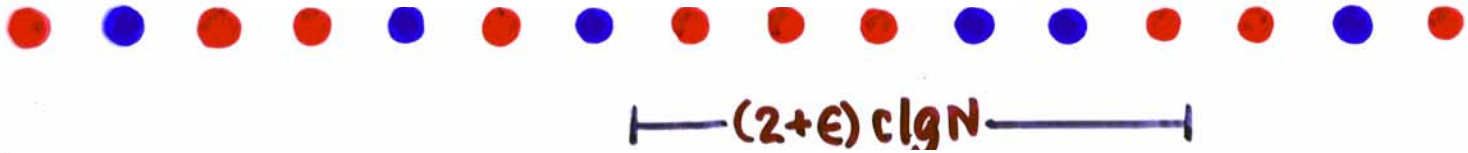


Claim: For sufficiently large c, in any window of size $(2+\epsilon)c \lg N$, there are $> c \lg N$ support elements and $< (1+\epsilon) c \lg N$ intercalated elements.

Recall: k support elements take space $(2+\epsilon)k$.

$\Rightarrow$ room for intercalated elements

Alternative Analysis



Claim: In any window of size $\Theta(\lg N)$ there are $\Theta(\lg N)$ support elements and $\Theta(\lg N)$ intercalated elements w.h.p.

Similar to # coin flips until we get a head.
$O(1)$ in expectation & $O(\lg N)$ w.h.p.

Coins are not independent, but negatively correlated.
Easy to solve using Chernoff bounds.
Can also solve directly using basic probability.

Concluding analysis

- Pr[given set C, $|C|=(2+\varepsilon)c \lg m$ has too few support elements]

$$\leq \sum_{j=0}^{c \log m} \binom{|C|}{j} \left(\frac{m}{2m-|C|+1}\right)^{j} \left(\frac{m}{2m-|C|+1}\right)^{|C|-j}$$

$$\leq \left(\frac{m}{2m-|C|+1}\right)^{|C|} \sum_{j=0}^{c \log m} \binom{|C|}{j}.$$

...which is polynomially small.

Minor Detail

Holds only when # elmts is large, but...

Claim: while the number of elements $k \le \sqrt{n}$, the total cost for library sort is $O(n)$.

$\Rightarrow$ only need to consider case $k \ge \Omega(\sqrt{n})$.

Thm: each insertion has cost $O(\lg N)$ w.h.p.

Gaps in My Knowledge

This talk: average-case analysis of naïve folk insertion.

- Related work: Ave-case priority queues
  [Itai,Konheim,Rodeh81]
  Contains most ideas of LibrarySort.
  LibrarySort simplifies.

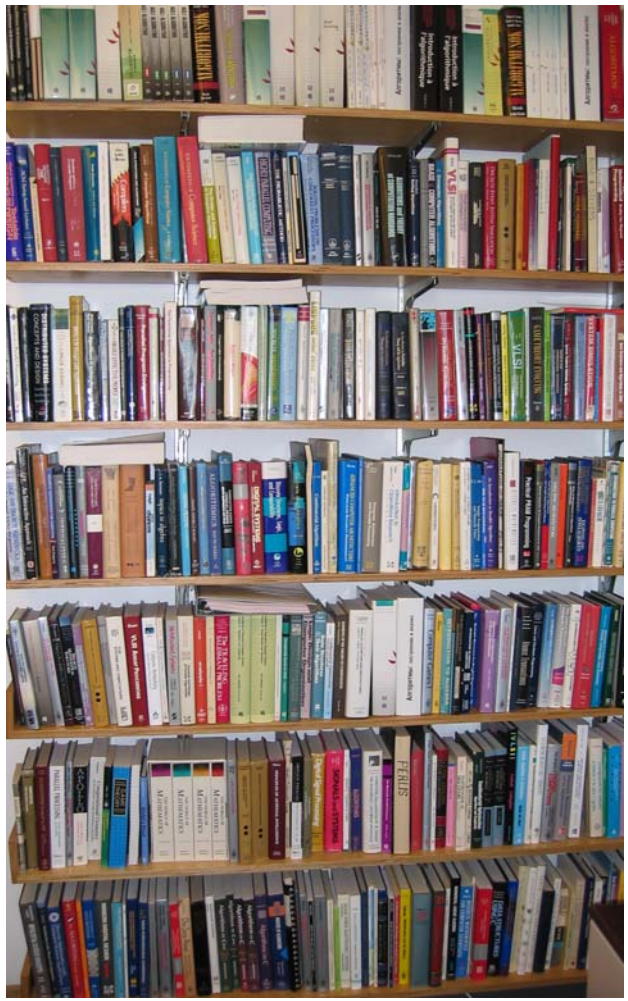
Gaps in My Knowledge

Other work: rebalance schemes for worst case.

	Upper bound	Lower bound
$O(N)$ gaps Sequential File Maintenance	$O(\lg^2 N)$ insert [Itai,Konheim,Rodeh] [Willard]	$\Omega(\lg N)$ insert [Dietz][Seiferas]
poly(N) gaps order maintenance, list labeling	$O(\lg N)$ insert [Deitz][DietzSleator][Tsakalidis][Bender,Cole,Demaine,FarachColton,Zito]	$\Omega(\lg N)$ insert [Dietz][Seiferas]
$O(N)$ gaps in external memory packed-memory structure in cache-oblivious algorithms	$O(1+\lg^2 N/B)$ insert	

Why do computer scientists say that insertion sort is $O(N^2)$?

Personal Experience?



Bookshelves of Distinguished Computer Scientists

Results

- **LibrarySort**, a natural implementation of **InsertionSort** with gaps.

- **Theorem:** **LibrarySort** runs in $O(N \lg N)$ time w.h.p. and uses linear space. Each insertion is exp $O(1)$ and $O(\lg N)$ w.h.p..